Using Spatial Techniques to Decrease Message Passing in a Distributed VE System

Olof Hagsand[*] Rodger Lea[†] Mårten Stenius [*]

Abstract

In an attempt to reduce the communication costs in a shared distributed virtual environment (VE) system with a large number of participants, we have explored the use of spatial interaction techniques. In particular, we have experimented with the notion of an area of interest (AOI) which is dynamically established based on the spatial proximity of participants in the shared space.

The work presented here describes an effort to integrate an enhanced aura-based spatial interaction model within the DIVE distributed Virtual Environment system and use this to reduce the overall communication load in the system. By associating lightweight communication groups to AOIs (area of interest), we show how messages are distributed to smaller peer groups, so that overall message passing is decreased. A multi-user application scenario is implemented as a case study. Real experiments and measurements with the current DIVE system using the scenario were made to validate the interaction model and its implementation.

This work was performed as a joint research effort between Sony CSL and SICS as part of the Wide Area Virtual Environment (WAVE) project.

Keywords: distributed virtual environments, 3D, multicast, VRML

1 Introduction

Multi-user, distributed VE systems are currently being designed and implemented that support large numbers of participants simultaneously connected to the global Internet. Such systems have the potential to offer an attractive infrastructure for a variety of collaboration applications including, traditional CSCW, multi-user gaming, and multi-user commerce applications. However, to gain wide spread acceptance within a non specialized community, the performance of such systems must be acceptable to general users[2].

At the gross level, the performance of such shared 3D virtual environments is based on two key aspects; the graphics performance available to render the 3D scene, and the communications performance required to maintain the shared scene and associated interaction information.

The cost of rendering complex 3D scenes with a large number of lines and polygons in real-time, includes 3D coordinate transformations, hidden surface removal, texture mapping, etc. However, rendering optimization, dedicated graphics hardware, level-of details (LODs), etc, can all be applied to single user systems which are essentially a "standard" 3D rendering problem.

In a distributed VE system, although rendering is a significant problem, the most intractable performance problem derives from the communication cost needed to maintain consistency between peers.

[*]Swedish Institute of Computer Science, Box 1263, S-164 28 KISTA, SWEDEN, email:olof@sics.se, mst@sics.se

[†]Sony Computer Science Lab. Inc. Tokyo, Japan, email: rodger@csl.sony.co.jp

1.1 A scalable infrastructure

A naive and basic infrastructure for a shared 3D world is simple; it consists of a database of objects that exist in the world, a set of tools to populate that database and a set of devices that display the contents of the database. The display device doubles as an input device and allows users to navigate through the world and to interact with other users and objects in the world. Updates to the world database by individual users are sent to the database which updates its internal model and informs other users of the changes. To achieve this, it requires some form of communication that will allow the display devices to access the database and to propagate user input to the database.

However, because we wish the system to scale, the use of a single shared database and a client-server model is not acceptable. Rather we adopt a peer to peer approach based on a replicated database. Each peer in the system holds a replicated copy of the "world" database and any changes to one copy have to result in updates to all other copies of the database.

1.2 Consistency in a distributed VE

The fundamental model presented by a distributed VE platform is one of a shared 3D space. Such a space, because it is shared, must be seen "consistently" by all users of that space. A system can provide different levels of consistency ranging from a strict interpretation to best effort.

In a strict interpretation, any actions that occur in the shared space must be propagated to all participants in that space, and conflicts between user actions are either avoided, or resolved. Furthermore, actions in the space maintain their causal relationship so that a user can make sense of a “happened before” and “happens after” relationship.

Obviously maintaining the consistency of the replicated database is a complicated task and one that requires significant exchange of information between peers. The choice of algorithm is crucial to the amount of message passing needed to reach consistency. Any distributed consistency algorithm has two major concerns:

- Membership: The membership of the consistency group, i.e, who is taking part in the consistency algorithm is crucial to performance. Any mechanism that reduces the number of participants in the consistency group directly reduces the number of messages that must be exchanged.

- Consistency guarantee: Once membership has been decided, the next issue is what model of consistency is used by the consistency algorithms. There has been much work in the research community addressing the issue of distributed consistency in more traditional data applications with a goal of reducing the cost of the algorithms. This work has concentrated on relaxing the degree of consistency either in a temporal domain[7, 12], or in a data value domain[19]. As part of the WAVE project we are addressing the issue of consistency in large VE’s, and in particular looking at weak and adaptive consistency based on group communications [13, 14].

In this paper we concentrate on the first of these issues, defining which participants must take part in any consistency algorithm. As mentioned above, the issue of membership is key to performance. If it is possible to find a mechanism or model that minimizes the number of participants in any given consistency decision, then we automatically reduce the communications overhead of the consistency algorithm.

Our approach has been to use group communications to support a spatial model based on the spatial domain that we target, shared 3D scenes. This feature is only present in shared 3D environments which are primarily concerned with supporting a notion of 3D space that models, to some extent, physical space, to allow us to reduce the participants in any consistency decision.

Assuming a notion of strict consistency between peers in the 3D virtual environment, then these peers must exchange information to ensure consistency of their spatial location and their interactions with other entities in the VE. As the number of users increase, the processing of incoming packets, and the operations associated with processing the incoming information, increases. If every peer emits an invariant rate of information, and network-supported multicast is used, the traffic and processing cost increases linearly with the number of peers. This is in agreement with our experiences with our experience with the DIVE system.

1.3 Spatial areas of interest

In previous experiments with the DIVE system we have observed that participants form sub-groups where activities occur in clusters or peer-to-peer within the global session. This observation can be exploited to decrease overall message passing if one can deliver packets only to the recipients they are intended for, i.e. those within the sub-group. In this way, the amount of global traffic is limited, and the number of incoming messages to each user is reduced.

Using the three dimensions of space is a well-known approach to partition VEs into several more or less disjoint areas of interests (AOIs). Static geographical regions are used in applications based on natural terrains, such as in DIS based systems[1].

A different approach uses intersecting volumes to model interaction between participants, so called spatial interaction models[4]. The simplest models use three-dimensional volumes, *auras*, to describe the spatial extent of an AOI. When two auras intersect, this is seen as an indication of mutual interest and thus a potential for the exchange of information.

The model goes further by defining two notions, *focus* and *nimbus*, to represent the degree of interest users have in each other. The focus represents the degree of interest a user brings to bear on another. The nimbus represents the degree of attention a user pays to another. The combinations of the focus and nimbus of two interacting users defines their level, or degree of interaction.

It is this model that we seek to use to drive our group communication mechanism and to reduce the number of participants in any consistency algorithm. In the rest of this paper we detail how we exploit this model and how we map it to group communication model.

In Section 2 the spatial model is presented; Section 3 presents light-weight groups in DIVE; Section 4 shows how the spatial model was implemented in DIVE using light-weight groups; Section 5 introduces the application scenario; and Section 6 describes the results of the experiments. In the next section, we briefly overview related work not already discussed in the main body of the text and finally, in Section 8 we conclude and discuss future plans.

2 Using auras to reduce communication

Our system model consists of a replicated database containing information about all objects in the shared virtual world. Each world defines a virtual space captured using a 3D co-ordinate system. Each object specifies a dynamic aura that represents the portion of the virtual space in which it is interested relative to itself. A separate unit, an **Aura Manager (AM)** constantly monitors objects as they move around the shared world and informs an object when other objects collide with its aura. The information from the AM to the object consists of a communications end point that allows a local replica for the remote object to be constructed locally by communicating with the remote object.

To ensure that the AM does not become a bottleneck in large scale systems we define a hierarchy of AM’s with a unique root, which also serves as a name server. This model allows us to share load by assigning AM’s to unique areas within the shared scene.

Figure 1 shows a simple two level AM hierarchy with each AM responsible for a single quadrant of the shared scene. The master AM (MAM) is used to handle AM handover as an object moves from one quadrant, and hence one AM, to another. Obviously, this hierarchy may be extended indefinitely.

In Figure 9, we introduce an example further explored in Section 5, the *avatar exhibit.* We can see a simple system with three user objects and two simple scenery objects (a grid and an ellipse on the grid). For the purposes of this discussion, scenery objects are always visible to all users of the system, i.e. every user holds a replica of a base database which includes scenery objects.

The wireframed spheres surrounding the three users are their auras, normally invisible. The two remote users are in each other’s auras and so holds information about each other. The third user, however, is not in the aura of the others, and therefore only needs to replicate the base database containing the scenery objects. It does not need to replicate the parts of the database that hold information about the other two users.

In essence, we use the notion of aura to partition the database, and the Aura Manager to track the database partitions.

The aura may be dynamic; for example, when a user enters a crowded room, then it is likely that the user would wish to reduce

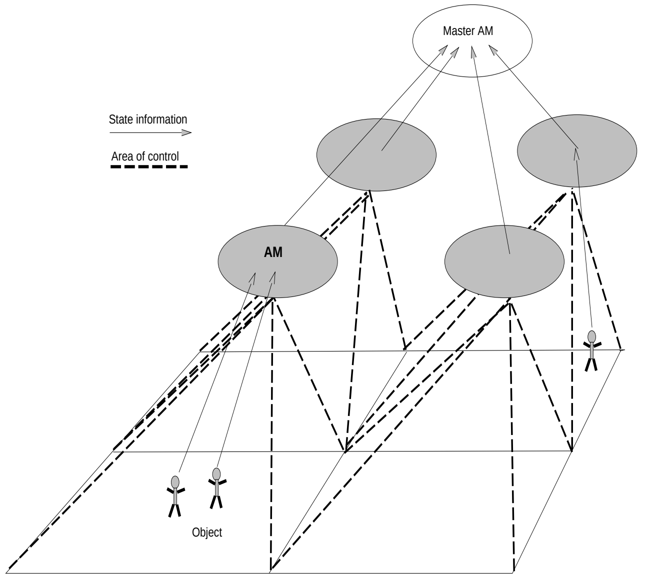

Figure 1: (Aura manager hierarchy)

their visual aura to cut down on the amount of information they need to be concerned with. This user notion maps directly to our requirements at the communications level where we wish to reduce the the degree of interaction to minimize the amount of consistency that must be supported.

The aura may also be dynamically adjusted to adapt to a specific environment. For example, if a user enters a room where the user should communicate with everyone in the room, the user's aura can coincide with the spatial boundary of the actual room. In this way, the auras of all users within the room will be aligned, and thus shared the same communication group.

Although the basic mechanism of the aura manager is similar to the MASSIVE approach, the resulting action is different. Since our system model relies on a replicated database model, we use the aura manager as a means to signal to the clients which parts of the database need to be replicated.

To do this, we rely on the underlying use of communication multicast groups. Parts of the replicated database are associated with multicast groups. When a client is informed of an aura collision it is passed a group identifier. The client then joins that group and requests the current state of the objects associated with that group. Once it has installed these objects in its copy of the replicated database, it then monitors that multicast group to track updates to that portion of the data base.

In the following sections we discuss the underlying group mechanism implemented in DIVE to support this model, and discuss in further detail how the aura manager mechanism is implemented using these light weight groups.

3 Light-weight groups in DIVE

DIVE has always used group communication based on multicast to support distributed state. In the original DIVE system, the group communication mechanism was based on Isis [5]. More recent versions of DIVE have used an SRM based distribution package [10], which implements a reliable multicast protocol using negative acknowledgement[8]. However, in all previous work, the granularity of the group has been large, eg a "world". In the work reported here, we have implemented a fine grained group mechanism that we refer to as "light weight groups".

3.1 Entities and groups

In DIVE, *entities* form the basic unit of information being addressed and distributed. Entities are hierarchically structured as trees, where *worlds* are roots, *nodes* contain structural information, and graphical *views* and *lights* form leaves. An entity hierarchy (with a world as root) defines a geometric space disjoint from other worlds. Worlds are associated with named multicast groups; peers obtain group addresses from a nameserver. Additionally, unnamed so called *light-weight* multicast groups can be associated with any entity. The two kinds of groups are, however, equal in the sense that they are both implemented by IP multicast and both are associated with entities.

The world group and the light-weight groups form a hierarchy of groups, much in the same way as entities form trees. We say that a group *encloses* an entity (or another group) if the group is the closest group "above" the entity in the hierarchy. That is, the first group that is encountered when going upwards in the entity hierarchy. Messages sent to an entity are always sent over the entity's enclosing group.

In the case of light-weight groups, any peer joining a group of an entity must also join the entity's enclosing group. The reason for this is that the entity hierarchy defines the context of an entity, so that an entity is well-defined only in the presence of its ancestors.

Entity hierarchies are *partially* replicated among peers. That is, peers may chose not to cache parts of the complete entity structure, due to parts of the world scene being far away or contain complex data. When the positions or surroundings change, non-cached entities can be requested by sending a request message. Here, we only treat the case where light-weight groups define a fully replicated sub-hierarchy. That is, a sub-hierarchy that is not replicated at all peers must have an associated light-weight group.

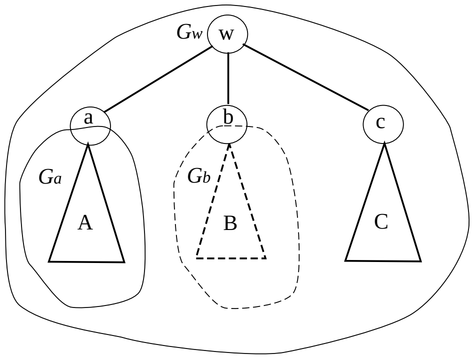

Figure 2: Partial replica of entity hierarchy as instantiated in one peer.

Figure 2 shows an instantiation (a replica) of an entity hierarchy as seen from one peer process P. The tree has a world w as root, a, b, c are nodes, and A, B, C are sub-trees containing nodes and leaves (possibly defining avatars). For the purpose of the example, a and b can be considered to be users (actors); with their subs defining their avatars. Actor a, resident in P, is represented by an avatar given by A; b represents a remote actor resident in some other peer Q with avatar B; and C represents pure world data. a and b define light-weight groups with associated groups G_a and G_b, respectively. G_a encloses A and G_w encloses a, b, c and C. Messages to an entity in A are sent over G_a, messages to B are sent over G_b, while messages to a, b, c, w and C are sent over G_w. In the figure, the triangle containing B is dashed indicating that P has not joined G_b and has no replica of B. [1]

Note that messages sent to an entity with a light-weight group are not sent over that group, but rather over the enclosing group. In this way, entities with light-weight groups are dual: they can be seen as members of two groups, the enclosing group and its own light-weight group. Thus, entities with light-weight groups serve as a "glue" between groups.

3.2 Dynamics of light-weight groups

When a peer joins the world group, it receives replicas of the world data by state transfer from an already connected peer. However, only the "stubs" of entity hierarchies enclosed by light-weight groups are transferred initially. To fetch the group data, the light-weight group is joined and the enclosed data is explicitly requested over the group. In Figure 2, process P has not joined G_b, and does not replicate B, only the "stub" b. This is experienced by the fact that a can not see b's avatar, B.

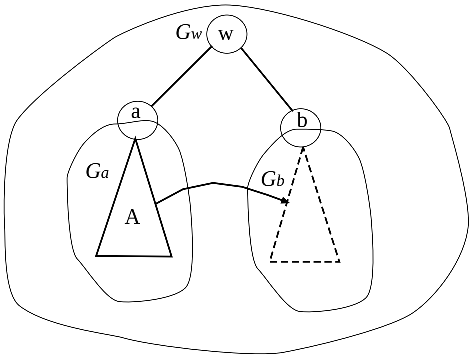

Figure 3: Moving a sub-structure A from one enclosing group to another.

Prune and graft operations on the entity hierarchy can be made easily even in the presence of light-weight groups. If a sub-structure is pruned from one part of a tree and grafted at another place, the enclosing group is simply changed and the graft operation is distributed to both groups involved. Figure 3 shows the example of an object hierarchy A originally enclosed by G_a, but moved to where it is enclosed by G_b. The notification of the operation will be distributed on both G_a and G_b, peers that have not joined G_b will "drop" A while peers that have joined G_b but not G_a will request (and make a replica of) A.

Note that the prune and graft operations are generalized to the case of separate worlds.

[1]The example is a simplification for several reasons: in DIVE, actors and avatars are different objects, and world data generally contains a large data set.

3.3 Exploiting light-weight groups

The light-weight group mechanism provides a basic service level that can be exploited on an application and server basis. In the case of the example in Figure 2, the decision and the actions to join G_b and request b's avatar, B, must be explicitly requested by P over G_b.

The issue of what criterion to use when requesting data and joining light-weight groups is not defined by the DIVE system. Different semantics can therefore be deployed in accordance with the preferred model. For example, peers can choose to base the decision on visual range, so that all groups within the vicinity of an avatar are joined. Another approach is to have a collision manager that notifies the peers of mutual collisions.

The latter approach is taken in this paper, and we will in the next section elaborate more on the details. In other work, such as the default behaviour in the current DIVE system, our approach is to use the visual range as a base for requesting entities. The DIVE platform can seamlessly support any such model.

4 Avatars and light-weight groups

In this section, we discuss how the the light-weight group mechanisms can be used to model the aura model introduced in Section 2. To be concrete, we present one design alternative that will be used in the remainder of the paper.

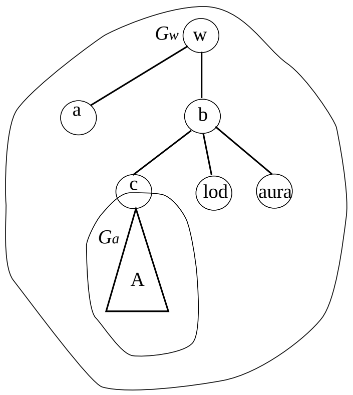

Figure 4: Entity hierarchy describing the design of an avatar, lod and light-weight group.

In the simplest form shown schematically in Figure 4, an actor (a) has an avatar (A), a simple representation of the avatar (lod), an aura ($aura$), and a light-weight group (G_a). Node b only acts as a "grouping" node. The avatar is enclosed by the light-weight group while the aura and lod is not. The simple movements of the actor are performed on object b, so that the movements of the aura and the lod is visible to every observer in world w. In contrast, motions and operations within the avatar A are relayed over G_a and are not visible to peers outside the group.

The 3D presentation of A, lod and $aura$ are shown in Figure 9. Three actors are shown: the left in low resolution (lod); the middle in full resolution with its surrounding aura; and the right in full resolution walking away from the booth. normally invisible. The simple lod representation of the avatar is a block-based representation only visible at a distance. Therefore, the full resolution of the avatar is not available to other actors until their auras intersect.

Additionally, the internal movements of the avatar, such as walking movements, is only visible within the light-weight group.

Given this structure of actors, the task of the Aura Manager is simply to track the auras of all actors, and to mutually notify them when their auras intersect. Two actors being engaged in an intersection mutually joins the light-weight group and requests the full avatar of the other actor. Thus, after aura intersection, the avatars of the actors are mutually visible.

The design in Figure 4 is used henceforth in this paper to model actors, avatars and auras. However, this design has one flaw: the simple actor movements are performed on b and are thus visible to the world group G_w. As the number of users increases, the traffic associated to these movements increases linearly. However, this traffic consists only of periodic position and velocity updates of b, which represent a low traffic in comparison with complex avatar motion. In any case, even this linear complexity can be removed by aura partitioning on a larger scale.

Another way to model the avatar is to enclose the complete actor (including aura and lod) by the light-weight group G_a. In this way, actors have no way of detecting other actors until their auras intersect. Only the aura manager has knowledge of the existence of all actor information. The problem with this approach is that a user has no overview of other avatars. As they get close, they just pop-up from nowhere.

We believe that the best approach is a combination of the two designs, where filtered motion (say once per second) update the lods that are only visible from a distance. In this way, a solution that scales arbitrary well is achieved, and you have the ability to have an overview of what other avatars are doing at a distance.

5 Application scenario: The avatar exhibition

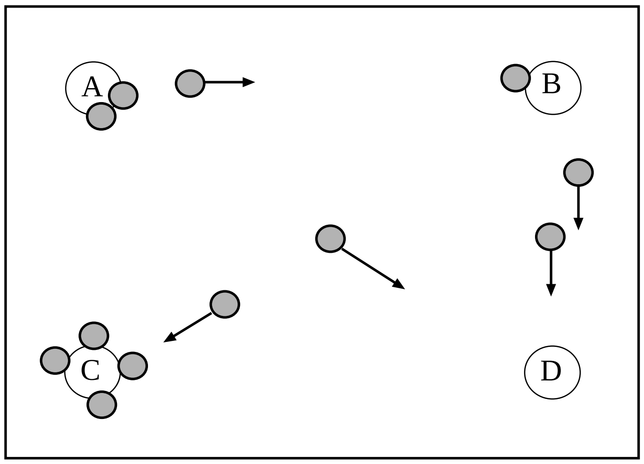

Figure 5: The Avatar exhibition scenario: A, B, C and D are exhibition booths, while the filled circles are visitors. Moving visitors are depicted with an arrow.

The *avatar exhibit* scenario simulates an exhibition situation where several users move between exhibit booths, where they form interacting subgroups, as depicted in Figure 5. We claim that this scenario is sufficiently general for a number of meeting applications where a large group of users is involved in (partitioned) communication.

The users are modeled by actors in a set labeled A. Each actor defines an aura, an avatar, a lod and a light-weight group as defined previously in Section 4. Initially, the only object an actor a has in its database, apart from the avatar and the background information (the exhibit area) are the lods and auras of the other actors. When a's aura collides with the aura of another actor, the aura manager notifies both actors. This results in a joining the light-weight group of the other actor and requesting its (full resolution) avatar. Symmetrically, the other actor joins the light-weight group of a and requests a's avatar.

Initially every actor randomly selects a booth to start from. It stays at this booth during a time interval T_B, and then moves to a new (random) booth during a time interval T_M, and then repeats the behaviour indefinitely. When moving, it moves at a constant speed as a series of positional updates and the limbs of the actor moves in quite complex actions mimicking human walking movements.

When visiting a booth, or when moving between them, an actor will meet other actors and then join the light-weight groups of the other actors.

The parameters of the scenario are the following:

- A - The set of actors - visitors at the exhibit.
- B - The set of booths.
- T_M - The time it takes an actor to move between two booths. Actors move with a constant speed, so T_M varies depending on the distance between booths.
- T_B - The time an actor stays at a booth. In the experiments, we let T_B be a stochastic variable.

6 Experiments and measurements

The main issue of the study, is to see how the system scales as the number of users increases. We would like to see the network traffic remain "constant" or increase very slowly when the population increases.

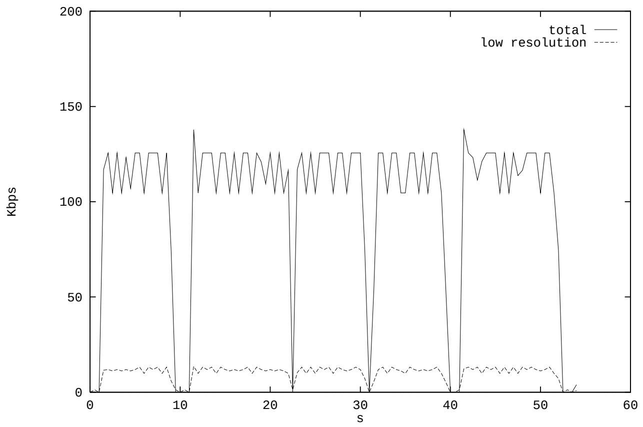

Figure 6: Three "walking" intervals of high-resolution data at 100Kbps and low resolution data at 10Kbps of a sending peer.

Our model is based on the assumption that actors communicate "locally" using a high-level of resolution, producing a high traffic load. This "high-fidelity" channel consists of actual walking movements animated using arms and legs. In the experiment, each peer produces a peak-rate of such traffic at a rate of 100 kbit/s. The other kind of traffic consists of "low-resolution" updates of positional and velocity updates. Typically, peak rate of such traffic is 10 kbit/s. See Figure 6 where the outgoing traffic of one participant has been measured during five walking intervals: the time at the booths can be seen as producing no outgoing traffic.

Our aim is to limit the high-resolution traffic to peers being in the spatial proximity. We therefore emit high resolution traffic on the light-weight groups associated with the auras, while the low-resolution data is sent to all participating peers within the exhibition.

As the number of users increases, however, so does the number of collisions between user auras. Therefore, we increase the number of booths linearly as well to keep the probability of encounters (from the viewpoint of each actor) at an approximately constant rate.

Loosely, the metric to measure is "network traffic". However, measuring the number of packets on the network depends on network topology, routing algorithms, etc. Therefore we have chosen to measure the number of *incoming* packets at each node and summarizing these numbers for all participating peers, and dividing this amount with $|A| - 1$. This amount is the mean incoming traffic per peer. and should, in the case of no light-weight groups, be equal to the sum of the sent traffic. When one global group is used, the mean incoming traffic should be equal to the sum of all sent packets. However, with light-weight groups, it should be significantly less. The closest measure of the sum of all sent packets ($< |A| * 110Kbps$), increases linearly with the number of participants.

In the experiments, we use a large number of Sun and Sgi computers connected by an heterogeneous LAN network (WAN experiments are being conducted but the results are not a part of this paper) with a minimum bandwidth of 10Mbps. The processes run a slimmed version of Dive3.1.0 with 3D rendering turned off.

In our experiments we have chosen to vary N between 5, 10 and 20, and to set B to 3, 5 and 10, respectively. T_M is approximately 10 seconds, and T_B is a stochastic variables with an average of 3 seconds. The number of collisions between actors are typically around 10 per actor and minute.

We also make a comparing study where all traffic is global, that is, disabling the light-weight group mechanism. In effect, this experiment was implemented by removing G_a from the avatar definition in Figure 4. In this way, all traffic in A is relayed over the world group, G_w. Since the peak rate is $110Kbps$ we have limited the number of peers in the global case to 10 so that we keep well below any congestion effects on the network.

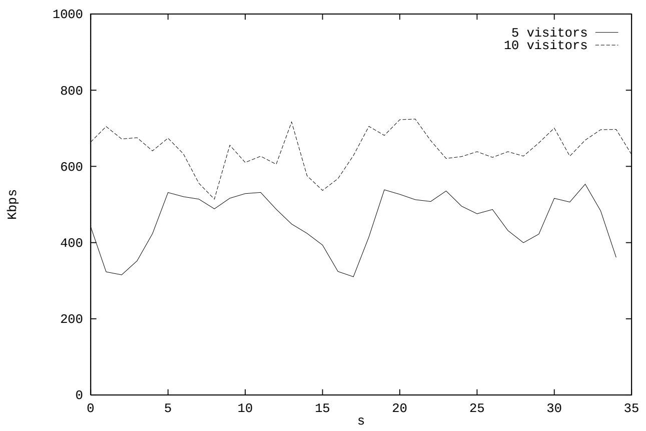

Figure 7: Mean total incoming traffic per peer with no light-weight groups for 5 and 10 peers.

6.1 The base case: no light-weight groups

In the first experiment, every peer receives all traffic from all peers. In Figure 7 the mean traffic received by each participants is depicted for the cases of 5 and 10 participants. The mean incoming rate is 470 Kbps for 5 participants and 643 Kbps for 10 participants.

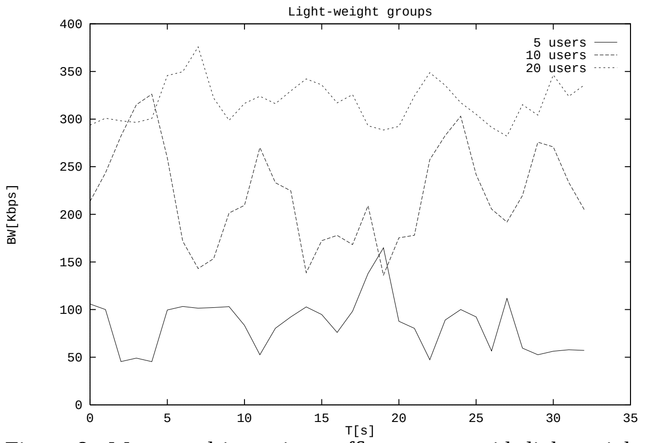

Figure 8: Mean total incoming traffic per peer with light-weight groups for 5, 10 and 20 peers.

6.2 Light-weight groups

In the second experiment, a participant receives high-resolution traffic only from those avatars whose auras intersect with the participants own aura. In Figure 8 the mean traffic received by each participants is depicted for the cases of 5, 10 and 20 participants. The mean incoming rate is 87 Kbps for 5 participants, 235 Kbps for 10 participants, and 342 Kbps for 20 participants.

6.3 Observations and discussion

In the case of the light-weight groups note that the lower limit of low resolution traffic is $10 * |A|$, that is 50, 100 and 200 Kbps. The traffic above this limit corresponds to high resolution traffic.

The receiving rate is less than the theoretical in the case when there is a lot of incoming traffic, for example, the experiments with 10 participants in Figure 8. We have identified the main reason for this behaviour to be the increased CPU load which is especially true for some of the low-end workstations used in the experiments (such as Sun sparcstation 1). This leads to (1) a decreased sending rate (2) packet loss in the receiving hosts input buffers, and thus an overall decrease in traffic.

In the experiments, we have not considered "state transfer" between peers, that is, the transfer of the initial state description between participants. Instead, each peer has a replica of every actor's full avatar a priori. However, in a real scenario, when a new avatar is first encountered, its state (graphical representation, etc) will have to be transferred which leads to a momentary peak of traffic corresponding to the size of the avatar description. This would lead to momentary peaks of traffic, not shown in the experiments.

7 Related work

There are several other projects that are looking at the issues of distributed VE's. For our purposes these can be classified into two types. The first type are those systems that are concentrating on supporting sophisticated group interaction models suitable for such work as CSCW. Such systems are characterized by interaction sessions in which participants act as both a source and a sink for updates. In addition, such systems tend to have stricter consistency requirements often involving locking or conflict resolution. The second class of systems are those that are interested in large scale distributed simulations where the main focus is on simple state update outside of any complicated interactions. Participants in these systems tend to act as sources of state updates. Although this is a false dichotomy in that there is a certain degree of overlap in any system, it allows us to usefully contrast approaches to sharing.

7.1 CSCW oriented systems

DIVE [10] is a sophisticated distributed VE developed at SICS and used for a number of CSCW projects [4]. The original work on spatial-based interaction models was carried out using DIVE.

There are several other distributed VE's that attempt to provide a framework for CSCW, including MR[20], Bricknet[21] and VEOS[6]. Four systems of note, because they implement some varient of a spatial interaction model are MASSIVE, CyberPassage, Aviary and Spline.

MASSIVE[9] has been designed to support shared conferencing in a 3D environment. MASSIVE only supports consistency on user objects, all other data, e.g. the world scene, is regarded as static background information. The MASSIVE system also uses the notion of auras but uses it to implement a spatial model rather than as we do, to define communication groups. In addition, MASSIVE does not use a group based communication model but relies on point to point links.

Aviary[23] although more concerned with immersive VR type applications and tightly coupled distributed platforms has a number of similar techniques to our system. In particular the use of the Environment Database (EDB) to manage collision detection and their model for splitting the EDB when loads are high is identical. However, Aviary has limited support for replication and uses a point to point communication model.

Spline [3] uses *locales* as a means to partition a virtual world. Communication within each locale is performed on separate multicast addresses. Each locale has explicit bordering information, defining how a locale relates to other locales and how a user can go from one to the next. A user listens to several locales simultaneously: the present and its neighbors.

The CyberPassage system[11] uses many of the ideas discussed in this paper. In particular, CyberPassage exploits the aura model for spatial partitioning. Since CyberPassage has been built from scratch to work with low cost PC machines and low bandwidth links, it implements a hybrid architecture that incorporates aspects of the client-server and the peer to peer model. In essence, the rendering machines (the home PC) is connected to a world server in a traditional client-server manner. However, the server is replicated in a peer to peer manner and uses some of the same low level mechanisms for group communication as those used by DIVE.

7.2 Distributed simulation platforms

NPSNET[15] is an example of the second class and one which has explored many of the issue of large scale interaction. Due to the target application set, distributed battlefield simulation, NPSNET has concentrated on different issue from our work. In particular, their main concern with respect to consistency is position updates of battlefield units.

NPSNET initially used “in-house” communication protocols, experimented with SIMNET[18] and finally adopted DIS[1]. In all cases, the underlying communication mechanism is a best effort broadcast with minimal consistency support. DIS includes support for position prediction algorithms, often referred to as dead reckoning[22] which offers a degree of consistency by predicting positional information based on movement physics.

Recent work by the NPSNET group has focussed on using multicast to address the scaling issue of the original broadcast approach of DIS[16] but again have taken a best effort approach to consistency. In the paper cited above, the authors discuss a possible Area of interest manager (AOIM) and propose a geographic approach to spatial partitioning mapping multicast groups hexagonal geographical regions. In simulation, using previous battlefield simulation data, they postulate that their approach will scale well.

7.3 Related work on the aura model

The aura based spatial model has been explored, to a certain degree, in earlier versions of DIVE [4, 10]. However, the notion of aura was treated orthogonally to the underlying database and used exclusively at higher levels to support interaction between users. In many cases, particularly those worlds that represent a large spatial area, such as a city, it is not necessary for a participant in one location to be consistent with another participant four blocks away. In that case, maintaining consistency at such a large granularity forces a high degree of false sharing [17].

MASSIVE has a different model of sharing to DIVE. MASSIVE only shares user information and does so through typed bidirectional links. MASSIVE explicitly tracks users and informs them of aura intersections. In this way, MASSIVE has a more explicit control over the communication links between users, but restricts itself to simply making communication endpoints available. There is no shared database as in the DIVE system.

The locales model in SPline [3] has many similarities with the light-weight group mechanisms described in this paper, specifically in how to bind multicast groups to object hierarchies. But the scope of locales is the same as spatial regions in NPSNET, although they are more flexible since any geometrical region can define a locale.

Our use of groups and multicast can be usefully contrasted with MASSIVE, SPLINE and NPSNET which have proposed mapping groups not to dynamic sets of related objects, but to spatial areas. NPSNET for example envisages using octagonal regions and assigning a multicast group to each region. Participants that enter a particular spatial area will join the associated group and receive all broadcasts to that group. The main drawback of this approach is that it suffers from both the scaling problem and the false sharing problem. In terms of scale, unless mechanisms exist for dynamic remapping of multicast groups to geographical areas, heavily used areas have no mechanism to reduce communication costs. In terms of false sharing, in a similar way to the original DIVE system, all participants in a geographical zone must participate in all consistency decisions.

MASSIVE uses the same model of aura as we do, even to a greater extent. However, the use of the spatial model has been used to drive point to point links rather than group protocols. The authors of the MASSIVE system imply that they will adopt multicast support in a similar manner to NPSNET, i.e. at a geographical level [9]. As such our comments on NPSNET apply equally here.

In contrast we have adopted an *object centric* approach to multicast groups based on a spatial aura and allow these auras to grow and contract according to application needs. In addition, our communication group model supports partial replication allowing further control over the false sharing problem (see Section 3.1).

Our goal in the work reported here was to use the notion of aura to drive the underlying replication mechanisms in a way that allowed us to keep the rich model of a fully replicated database, but without having to maintain total consistency between all replicas at all times.

8 Conclusions

As discussed in the introduction, the success of distributed VE's for widespread use rests on their ability to provide acceptable performance. Apart from graphics performance, the most significant cost in any distributed VE is the communication costs of ensuring that participants have a defined degree of consistency with respect the shared scene. The cost of maintaining consistency depends on two factors, the number of participants in the consistency decision, and the degree of consistency. In this paper we have attacked the first factor. We have shown how we have used a spatial model to decrease the number of participants in any consistency decision which is based on the notion of an aura or spatial area of interest. Further, we have shown how this spatial model can be used to drive a fine grained group communication mechanism allowing us to exploit multicast communication to further reduce message traffic. To support our claims for increased performance through reduced message traffic we have implemented a simple shared exhibition application and measured message traffic with and without the use of the spatial model. Our results show a dramatic decrease in message traffic arriving at a node.

The hypothesis of the experiments are based on the fact that multicast is "correctly" implemented in routers and computer network interfaces: First, the network interface must be able to handle the filtering of incoming multicast addresses in hardware. Otherwise, the protocol handling software will be forced to filter all multicast messages appearing on the LAN in software. Second, the multicast routing algorithms must be able to refrain from forwarding messages to networks where no member of that group exist. Both of these features must exist in order to benefit fully from the multicast results described. It is even more important in the WAN case, but as the Internet Mbone is deployed with multicast routers, we hope that more extensive use of multicast in the way described in this paper can lead to substantial performance benefits.

References

[1] DIS ANSI/IEEE std 1278-1993. Standard for information technology, Protocols for distributed interactive simulation. March 1993

[2] K. Arthur, K. Booth and C. Ware. Evaluating 3D task performance for fish tank virtual worlds. In ACM trans on distributed systems, 11(3):239-265 1993.

[3] J. Barrus, R. Waters and D. Anderson, "Locales: Supporting Large Multiuser Virtual Environments" IEEE Computer Graphics and Applications 16(6), Nov, 1996

[4] S. Benford, L. Fahlen, C. Greenhalge and J. Bowers. Managing mutual awareness in collaborative virtual environments. Proc. ACM SIGCHI conference on Virtual reality and technology (VRST'94) August 23-26th 1994, Singapore, ACM Press.

[5] K. Birman, A. Schiper and P. Stephenson. Lightweight causal and atomic group multicast. In ACM transactions on Computer Systems 9(3), 272-314. 1991

[6] W. Bricken and G. Coco. The VEOS project Presence. Vol. 3 No. 2. Spring 1994. pp. 111-129. MIT Press.

[7] J. Carter, J. Bennet and W. Zwaenepoel. Implementation and performance of Munin. Procs. of 13th Symposium on operating system principles (SOSP). Oct 1991 pp 152-164. ACM press.

[8] S. Floyd, V. Jacobson, C-G Liu, S. McCanne and L. Zhang, "Reliable Multicast Framework for Light-weight Sessions and Application Level Framing" Proceedings from SIGCOMM'95, Boston, MA, Sept, 1995

[9] C. Greenhalge and S. Benford. MASSIVE: a distributed virtual reality system incorporating spatial trading. Procs of the 15th ICDCS. May 30 - June 2, Vancouver Canada 1995. IEEE press.

[10] O. Hagsand. DIVE - A platform for Multi-User Virtual Environments. IEEE Multimedia Vol. 3. No. 1 1996 pp. 30-39

[11] Y. Honda, K. Matsuda, J. Rekimoto and R. Lea. Virtual society. Procs. of VRML'95, San Diego. USA. Dec. 1995 Available at: http://www.csl.sony.co.jp/project/VS/VRML95.ps.Z

[12] P. Hutto and M. Ahamad. Slow memory: Weakening consistency to enhance concurrency in distributed shared memories. Procs. of the 10th ICDCS. pp. 302-311 May 1990 IEEE press.

[13] R. Lea and Y. Yokote. Adaptive operating system design using reflection. Procs. of the 5th Workshop on Hot Topics in Operating Systems (HTOS-V). Orcas Island Washington, USA.1995. pp. 95-101. IEEE press.

[14] R. Lea, P.G. Raverdy, Y. Honda, and K. Matsuda. Issues in the design of a large scale VE. Sony Computer Science Lab Tech Report 95 available from http://www.csl.sony.co.jp/

[15] M. Macedonia, Pratt, D. and Zyda, M. NPSNET: A network software architecture for large scale virtual environments. Presence. Vol. 3 No. 4. Fall 1994. pp. 265-287. MIT Press.

[16] Macedonia, M., Zyda, M., Pratt, D., Brutzman, D. and Barham, P. Exploiting reality with multicast groups. IEEE Computer Graphics, Vol.15 No.5 pp. 38-45. September 1995. IEEE press.

[17] Mosberger, D. Memory consistency models Operating Systems Review Vol. 27. No. 1 pp. 18-26. ACM press.

[18] Pope, A, The SIMNET network and protocols. BBN report no. 7102. BBN systems and technologies, Cambridge, MA. USA. 1989.

[19] Pu, C. Relaxing the limitations of serializable transactions in distributed systems. Proceedings of the 5th ACM European Workshop, Le Mont St Michel, France. Operating Systems Review Vol. 27. No. 2 pp. 66-71. ACM press.

[20] Shaw, C., Green. M., Liang, J. and Sun, Y. Decoupled simulation in virtual reality with the MR toolkit. ACM trans. on Information Systems. 11(3), pp. 287-317.

[21] Singh, G., Serra, L., Png, W. and Ng H. Bricknet: A software toolkit for networked virtual worlds. Presence. Vol. 3 No. 1. Winter 1994. pp. 19-34. MIT Press.

[22] Singhal, S. and Cheriton, D. Exploiting position history for efficient remote rendering in networked virtual reality. Presence. Vol. 4 No. 2. Spring 1995. pp. 169-193. MIT Press.

[23] Snowdon, D. and West, A. AVIARY: Design issues for future large scale virtual environments. Presence. Vol. 3 No. 4 Fall 1994. pp. 288-308. MIT press.

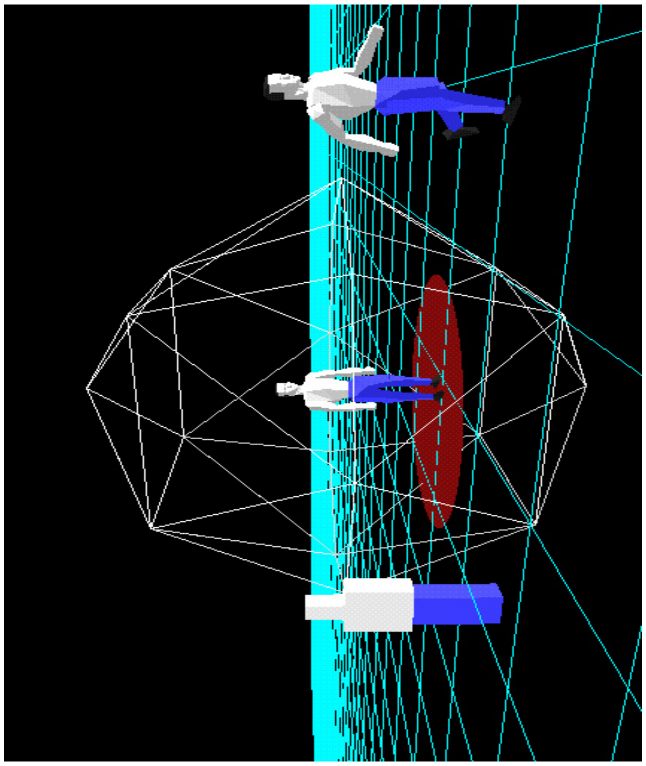

Figure 9: Three users with auras on a grid around a booth.